



Merlin Systems Corp. Ltd

**Miabot BT PRO
Sonar Array**

v1.0

Revision History

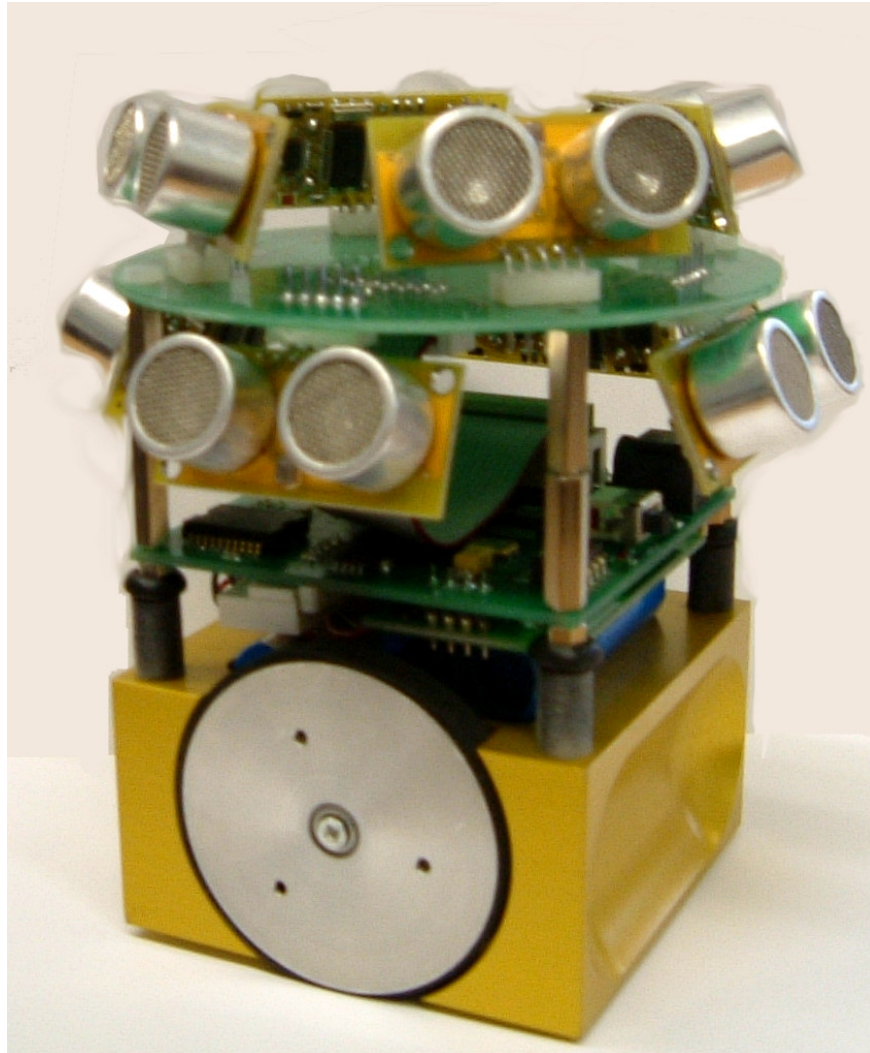
V1.0 07/03/05 pp first version

© Merlin Systems Corp. Ltd 2002-2004

Merlin Systems Corp. Ltd assumes no responsibility for any errors which may appear in this manual, reserves the right to alter the devices, software or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. Merlin Systems Corp. Ltd's products are not authorized for use as critical components in life support devices or systems.

Introduction

The Miabot Pro sonar array package comprises a pro robot, a 360° sonar array add-on for the top of the robot, and a demonstration PC application.



NOTE: This applies only to Miabot BT PRO robots, software v2.3 and above.

Sonar Array Hardware

The sonar robot add-on can support up to 8 miniature sonar rangefinder units covering the full 360° around the robot at 45° intervals.

In practice 1, 2, 4 or 8 units are commonly fitted, depending on the application.

The rangefinder devices are the Devantech (Robot Electronics) type SRF08.

They have a range of up to 6 metres and a half-angle of view of about 30°.

See the separate manufacturers datasheet for more information ("SRF08 Ultra sonic range finder.htm", on the support disk).

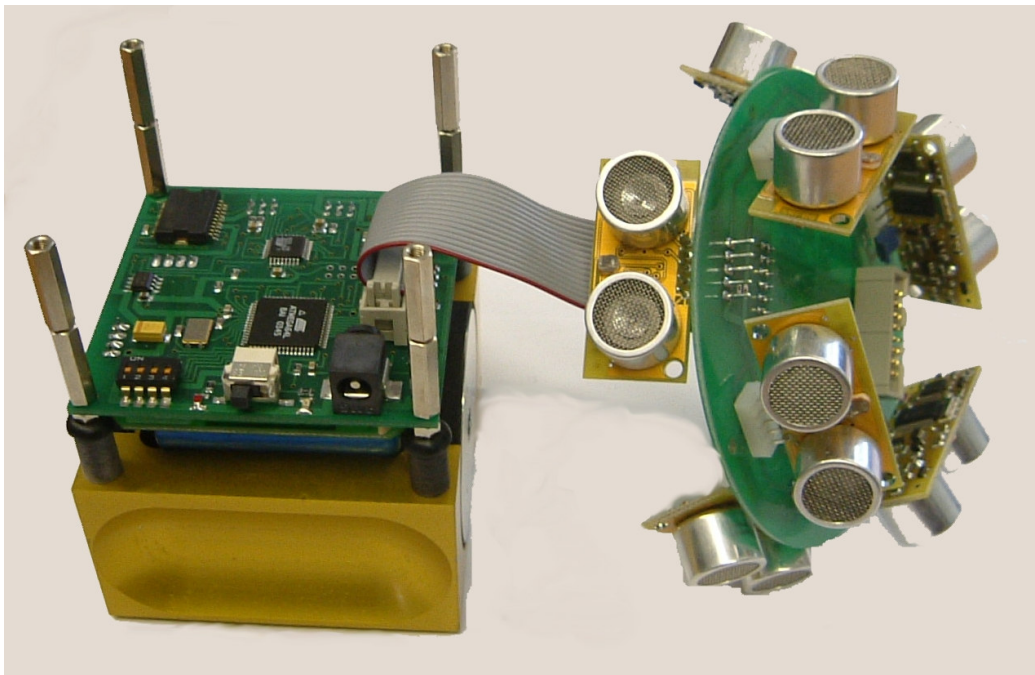
The rangefinders are connected to the robot via a common I²C bus. Robot commands to access the I²C bus are described in a separate section below.

Fitting

The sonar array fits over the main control board on the Miabot Pro.

The ribbon cable lead plugs into the Pro expansion connector.

For physical support, double hex pillars fit in place of the four corner screws securing the control board, and the sonar array board is screwed to the top of these using the original corner screws :-



The board orientation is such that the expansion connector on the sonar board fits directly above the one on the main board.

All fittings are M3 size.

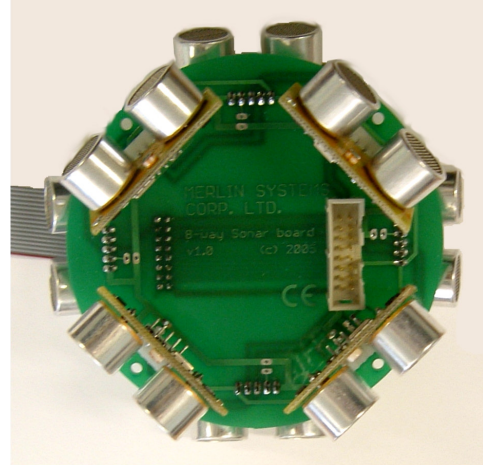
NOTE: If desired, the sonar board will fit on *single* hex spacers, but in this position the on/off switch and the charge socket are obscured.

Sonar Addresses

Each unit is given a separate bus address.

As supplied, the addresses for the individual sonar directions are set up as follows :-

		front		
		E0		
	EA		E8	
left	E6			E4 right
	EE		EC	
		E2		
		back		



These addresses are 8-bit hex values (the actual 7-bit I²C address is the upper 7 bits). This arrangement is assumed by the demonstration software.

To change unit addresses, refer to the manufacturer datasheet, and notes below on the robot I²C control commands.

You need to send a set of commands similar to –

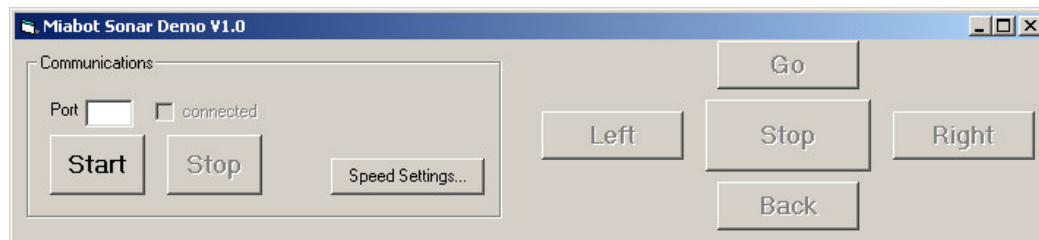
[iE0=0,A0] [iE0=0,AA] [iE0=0,A5] [iE0=0,NN]

A unit can be temporarily disabled by bridging the two-pin jumper installations next to its connector on the board: This can be useful in resolving bus address conflicts.

Demonstration Software

A simple example Visual Basic application is provided, which can be installed from the Miabot Support Disk “Sonar Array\Demo App” subdirectory. The source code is also provided.

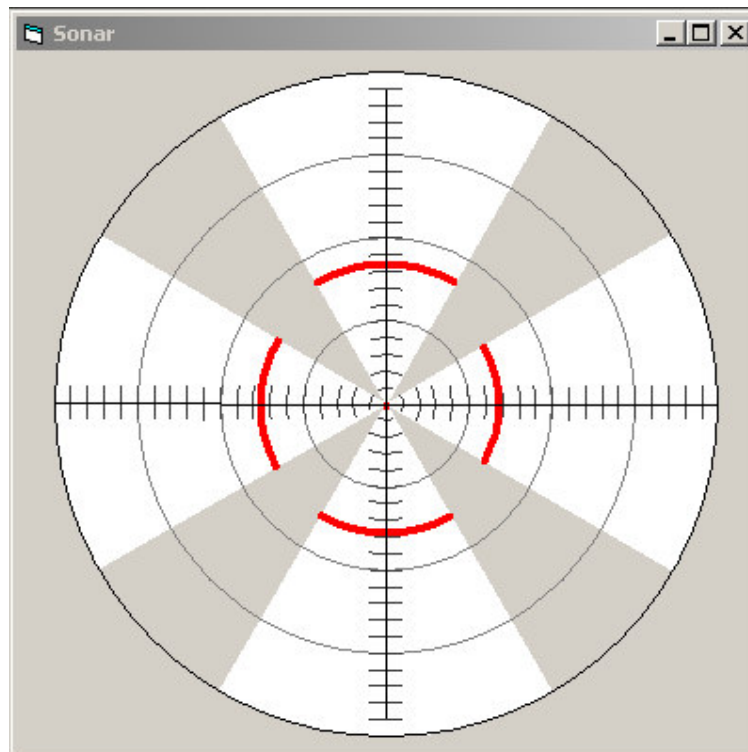
When run, this presents a window like this :-



Type the Bluetooth virtual Comport number into the 'Port' box, and hit the 'Start' button to start communication with the robot.

(The 'Stop' button on the left will stop communications and close the port).

At this point, the the application interrogates the attached sonars to find out which units are present, and will present a 1-, 2-, 4- or 8-way 'map' in the radar screen window, like this :-



(this is a 4-way example)

The 'radar' screen shows the sonar output as a series of segments with red arcs showing the echo returns.

The buttons on the right can now also be used to steer the robot around.
The “Speed Settings” button brings up controls to change the movement speeds.

Internally, the example application recognises specific 1-, 2-, 4- and 8-unit patterns: Sonars are arranged into 'groups' that are fired simultaneously, and these groups are operated in turn, to avoid cross-echoes between devices. It may be necessary to experiment with these arrangements to get the performance needed in a particular application.

The arrangement of the sonar units is assumed to match the diagram above.

The distance range of the display and the number of echoes displayed from each unit can easily be adjusted by editing the application.

At present, the application sets each sonar unit to 'minimum gain' on startup. In addition to this, it is also possible to reduce the scan range of the sonar, which can speed up the scan (see manufacturers datasheet).

Miabot Pro Commands for I²C Support

From version 2.3 onward, the Miabot Pro command-set contains a command to control slave devices on the the I²C bus, as follows :-

[i]-access I²C bus device

Two command formats allow reading and writing over the I²C bus :

[iAA=x1,x2...]

write data bytes (hexadecimal values x1,x2 etc.) to the bus at address 'AA'.

[iAA?nn]

read nn (decimal) bytes from the bus at address 'AA'.

The bus addresses 'AA' are hexadecimal values, of which the top 7 bits are the actual I²C address and the lowest bit must always be 0 (i.e. even numbers only).

Accesses with no data transferred are permitted - i.e. read or write 'no bytes'.

These are specified as **[iAA=]** and **[iAA?]**.

If an error occurs (including no bus response), the access will be retried a set number of times. If the maximum retries have occurred without a response, the command fails with an error code. (See below for retry controls and error codes).

All 'i' commands eventually produce a response string, but this may not occur for some time if a lengthy retry is specified. The following are the possible response formats :-

<i>

a write access succeeded (N.B. also read access with no bytes)

<i x1 x2 ...>

a read access returned the (hexadecimal) bytes shown (x1, x2...)

<i=nn>

the access failed, with error-code nn

Current error-code values are -

<i=1> access timed out: no response after N retries (see below)

<i=2> access rejected: expected acknowledge bit missing (during write only)

All I²C accesses are automatically retried, according to two parameter settings -

[.iN] number of bus-access retries -can be zero. Default = 3

[.iT] interval between bus-retries (in milliseconds)

Examples:

[iC4?3]	read three bytes from I ² C address 0x62	-->	response <i 25 A7 03>
[iA0=1,21]	write two bytes to address 0x50	-->	response <i>
[i70?14]	read 14 bytes from 0x38 (no device)	-->	response <i=1> (failed)

Commands to Control Sonars

To update and read results from a sonar, three accesses are required each time (see datasheet for full explanation) :-

- (1) First, send a ranging (or 'ping') command to start the unit scanning
e.g. “[iE0=0,51]”
- (2) Second, set the internal access address to '02' to access the correct returns data
e.g. “[iE0=2]”
- (3) Finally, read out the echoes data
e.g. “[iE0?10]” (to read back the first 5 echoes as 5* byte-pairs)

The ranging operation itself can create a considerable delay (up to 65 milliseconds), during which the sonar device will not respond to I²C accesses at all.

The example application uses the inbuilt retry controls to detect when the new sonar information is ready. To do this -

- the retry controls are set to “[iT=5]”, “[iN=20]” to retry at 5mS up to 100mS

- *first* all 'ping' commands in a group are sent

- next the application sends the 'pointer=02' commands to each sonar.

The first of these will be held up retrying until the first sonar scan completes.

(The other scans will normally complete at roughly the same time, so will not cause further delays).

- finally, results are read out from each unit in turn